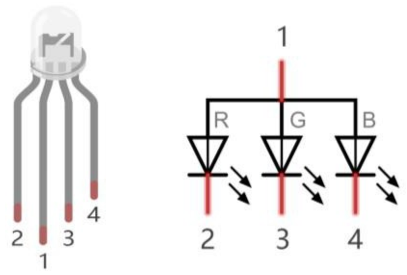
Lesson 6 How to light the ws2812 led

6.1 Knowledge of ws2812 led

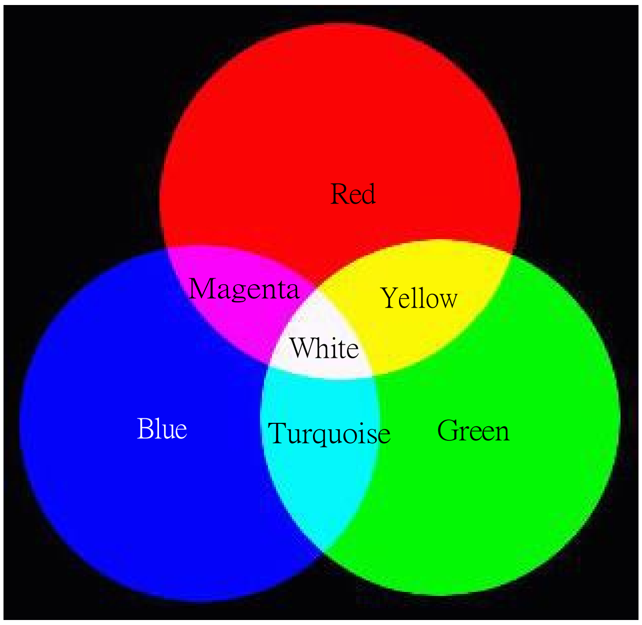
WS2812B is a intelligent control LED light source that the control circuit and RGB chip are integrated in a package of 5050 components.It internal include intelligent digital port data latch and signal reshaping amplification drive circuit.Also include a precision internal oscillator and a voltage programmable constant current control part, effectively ensuring the pixel point light color height consistent.
The data transfer protocol use single NZR communication mode. After the pixel power-on reset, the DIN port receive data from controller, the first pixel collect initial 24bit data then sent to the internal data latch, the other data which reshaping by the internal signal reshaping amplification circuit sent to the next cascade pixel through the DO port. After transmission for each pixel, the signal to reduce 24bit. pixel adopt auto reshaping transmit technology, making the pixel cascade number is not limited the signal transmission, only depend on the speed of signal transmission.
RESET time>280μs , it won't cause wrong reset while interruption, it supports the lower frequency and inexpensive MCU.
Refresh Frequency updates to 2KHz, Low Frame Frequency and No Flicker appear in HD Video Camera, it improve excellent display effect.

LED with low driving voltage, environmental protection and energy saving, high brightness, scattering angle is large, good consistency, low power, long life and other advantages. The control chip integrated in LED above becoming more simple circuit, small volume, convenient installation.
A RGB LED has 3 LEDs integrated into one LED component. It can respectively emit Red, Green and Blue light. In order to do this, it requires 4 pins (this is also how you identify it). The long pin (1) is the common which is the Cathode (+) or positive lead, the other 3 are the Anodes (-) or negative leads. A rendering of a RGB LED and its electronic symbol are shown below. We can make RGB LED emit various colors of light and brightness by controlling the 3 Anodes (2, 3 & 4) of the RGB LED



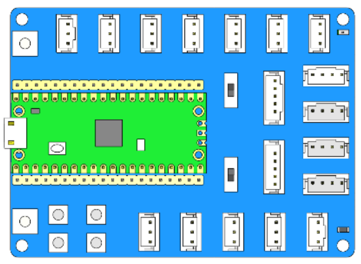
Red, Green, and Blue light are called 3 Primary Colors when discussing light (Note: for pigments such as paints, the 3 Primary Colors are Red, Blue and Yellow). When you combine these three Primary

Colors of light with varied brightness, they can produce almost any color of visible light. Computer screens, single pixels of cell phone screens, neon lamps, etc. can all produce millions of colors due to this phenomenon.

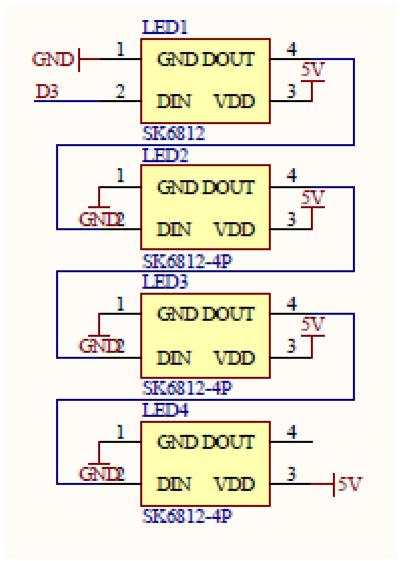


We know from previous section that, control board controls LEDs to emit a total of 256(0-255) different brightness with PWM. So, through the combination of three different colors of LEDs, RGB LED can emit 256^3=16777216 Colors, 16Million colors.

6.2 Components & Parts

Components	Quantity	Picture
Raspberry Pi Pico	1	
USB cable	1	
Pico Expansion board	1	

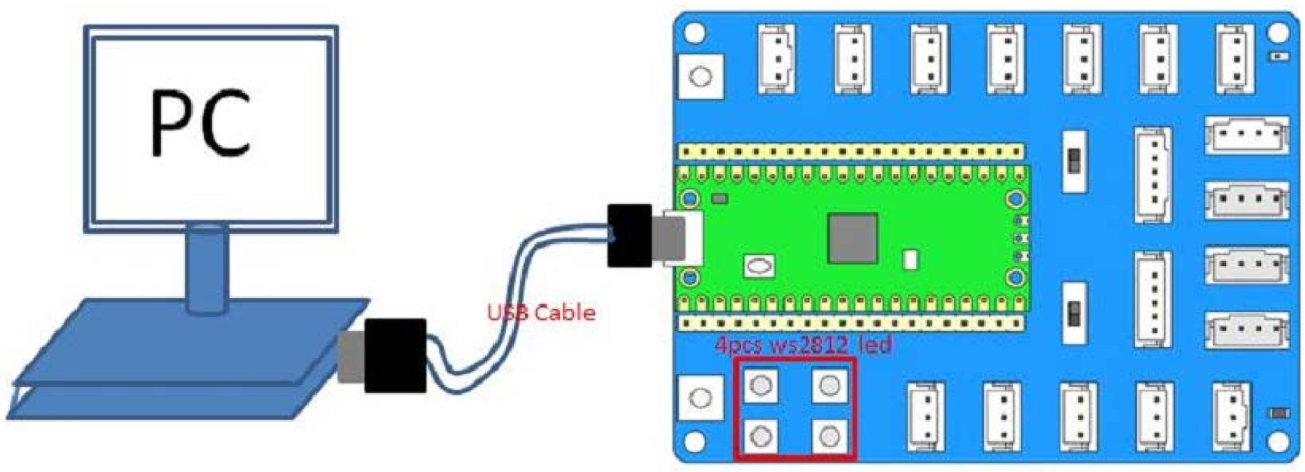
6.3 Experimental Principle



This course controls the ws2812 led occupying the GPIO3 pin on the Pico Expansion board by programming the Raspberry Pi Pico.

6.4 Circuit

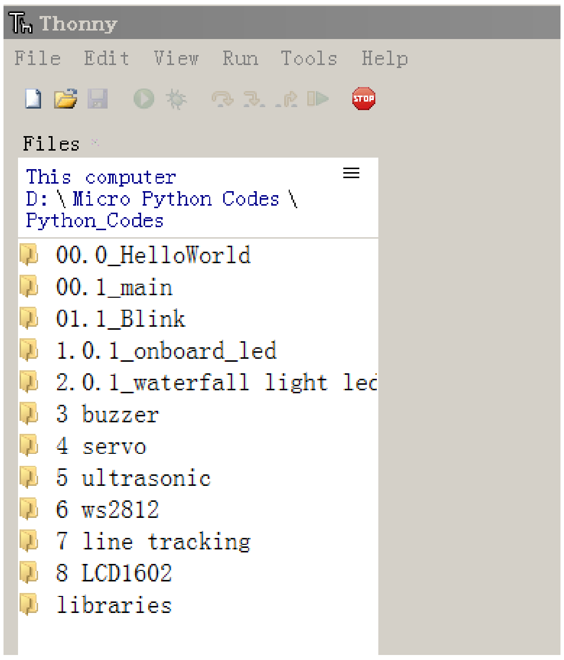
Please refer to the picture below to install the Pico on the expansion board. Note that the USB port of the Pico should be in the same direction as the power port of the Pico Expansion board. After installation, connect the Pico to the computer via a USB cable.



6.5 Run the program

6.5.1 Codes used in this tutorial are saved in "Pico Expansion Kit Tutorial\Lessons for Python\Python_Codes". You can move the codes to any location. For example, we save the codes in Disk(D) with the path of "D:/ Micro Python codes".

6.5.2 Open "Thonny", click "View"--"Files"--"This computer"--"D:"--"Micro Python Codes"--"Python_Codes".

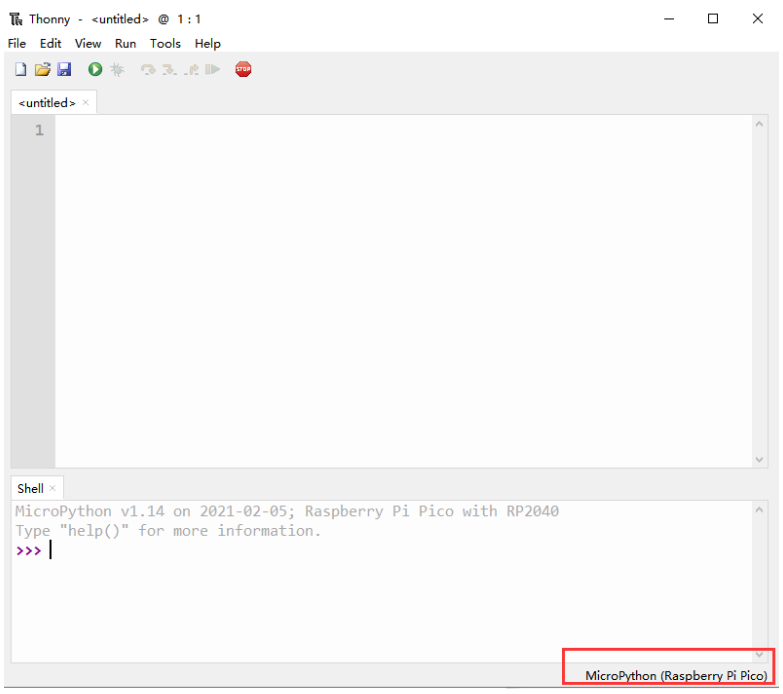


```
This computer
D: \ Micro Python Codes \
Python_Codes
00.0_HelloWorld
00.1_main
01.1_Blink
1.0.1_onboard_led
2.0.1_waterfall light led
3 buzzer
4 servo
5 ultrasonic
6 ws2812
7 line tracking
8 LCD1602
libraries
```

This computer: The file area of the personal computer.

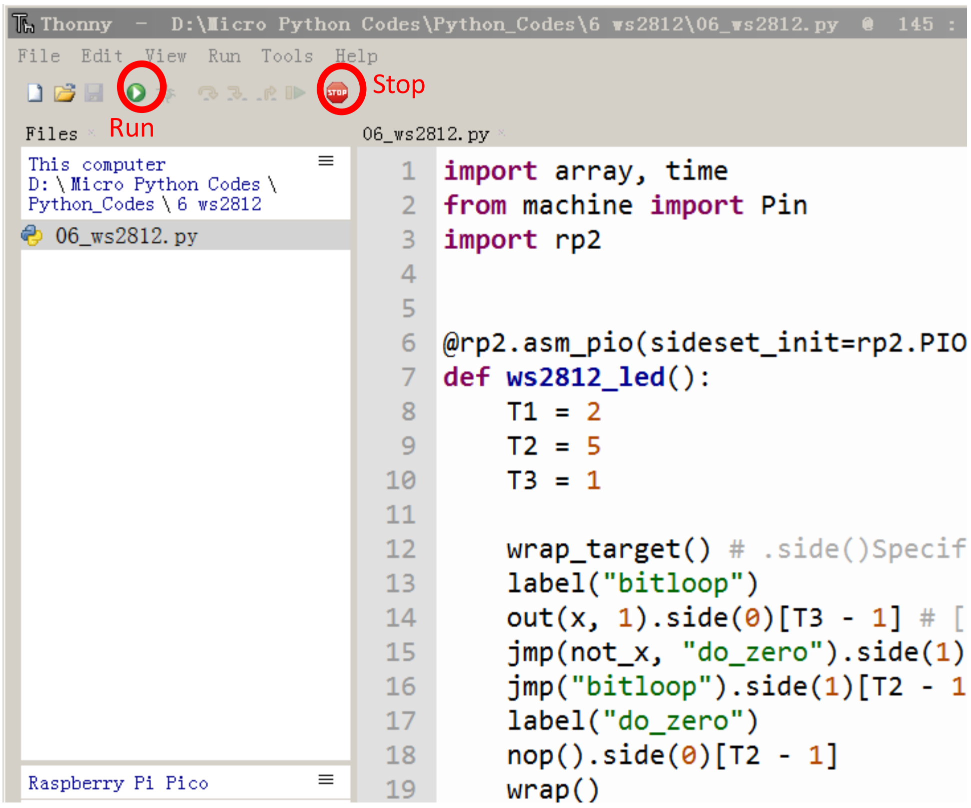
Raspberry pi Pico: Pico file area, the code saved in Pico can be viewed in this area.

6.5.3 Confirm that "MicroPython (Raspberry Pi Pico)" is displayed in the lower right corner. If it is not, please click the font in the lower right corner to select "MicroPython(Raspberry Pi Pico)" mode.


Thonny - <untitled> @ 1 : 1
File Edit View Run Tools Help
<untitled>
Shell
MicroPython (Raspberry Pi Pico)

```
MicroPython v1.14 on 2021-02-05; Raspberry Pi Pico with RP2040
Type "help()" for more information.
>>> 
```

6.5.4 Double-click the code “06_ws2812.py” required for this course. The content of the code will be displayed in the interface on the right.



```
import array, time
from machine import Pin
import rp2


@rp2.asm_pio(sideset_init=rp2.PIO
def ws2812_led():
    T1 = 2
    T2 = 5
    T3 = 1

    wrap_target() # .side()Specif
    label("bitloop")
    out(x, 1).side(0)[T3 - 1] # [
    jmp(not_x, "do_zero").side(1)
    jmp("bitloop").side(1)[T2 - 1
    label("do_zero")
    nop().side(0)[T2 - 1]
    wrap()
```

6.5.5 Click the Run button to run the program,then the 4 WS2812 LED on the Pico Expansion board will light in the order:red-- yellow-- green-- blue-- dark blue-- pink-- white--off

6.5.6 Click the stop button to stop the program,the 4 WS2812 LED on the Pico HAT board will light off.

6.5.7 Also,you can upload the "06_ws2812.py" and "main.py" to the Pico. Then you can run the program offline,disconnect the USB, and power on the Pico Expansion board ,you will see the 4 WS2812 LED on the Pico HAT board light in the order:red-- yellow-- green-- blue-- dark blue-- pink-- white—off. Turn off the power,Pico HAT board and 4 WS2812 LED stop working.

6.6 Code

06_ws2812.py

```
import array, time
from machine import Pin
import rp2


@rp2.asm_pio(sideset_init=rp2.PIO.OUT_LOW, out_shiftdir=rp2.PIO.SHIFT_LEFT,
autopull=True, pull_thresh=24)
def ws2812_led():
    T1 = 2
    T2 = 5
    T3 = 1

    wrap_target() # .side()Specifies where the program continues execution.
    label("bitloop")
    out(x, 1).side(0)[T3 - 1] # []Delay a certain number of cycles after executing the
instruction.
    jmp(not_x, "do_zero").side(1)[T1 - 1]
    jmp("bitloop").side(1)[T2 - 1]
    label("do_zero")
```

```
        nop().side(0)[T2 - 1]
        wrap()

class WS2812():
        def __init__(self):
                # Configure the number of WS2812 LEDs, pins and brightness.
                # define WS2812/SK6812(color_led) LED number, pin and brightness。
                self._led_num = 4 #    LED number
                self._pin = 3       # GPIO3
                self.brightness_value = 0.2    # light brightness. The value range is 0~1.
                # Create the StateMachine with the ws2812 program, outputting on
Pin(PIN_NUM).
                self.sm = rp2.StateMachine(0, ws2812_led, freq=8_000_000,
sideset_base=Pin(self._pin))
                # Start the StateMachine, it will wait for data on its FIFO.
                self.sm.active(1)
                # Display a pattern on the LEDs via an array of LED RGB values.
                self.ar = array.array("I", [0 for _ in range(self._led_num)])

        def pixels_show(self):
                dimmer_ar = array.array("I", [0 for _ in range(self._led_num)])
                for i,c in enumerate(self.ar):
                        r = int(((c >> 8) & 0xFF))
                        g = int(((c >> 16) & 0xFF))
                        b = int((c & 0xFF))
                        dimmer_ar[i] = (g<<16) + (r<<8) + b
                self.sm.put(dimmer_ar, 8)
                time.sleep_ms(10)

        # Set multiple colors
        # Set the color of the specified ws2812.
        def pixels_set(self,i, color):
                R = round(color[0] * self.brightness_value)
                G = round(color[1] * self.brightness_value)
                B = round(color[2] * self.brightness_value)
                self.ar[i] = (G<<16) + (R<<8) + B
                self.pixels_show()

        # Set the color of all ws2812 led.
        def pixels_fill(self,color):
                for i in range(len(self.ar)):
                        self.pixels_set(i, color)
```

```
    # Set a single color.
    def set_color(self,i,R,G,B):

        R = round(R * self.brightness_value)
        G = round(G * self.brightness_value)
        B = round(B * self.brightness_value)
        self.ar[i] = (G<<16) + (R<<8) + B
        self.pixels_show()

    def set_color_all(self, R, G, B):
        for i in range(len(self.ar)):
            self.set_color(i, R, G, B)

    # Adjust the brightness, the value range is 0-1, the default value is 0.2
    def brightness(self, brightness = None):
        if brightness == None:
            return self.brightness_value
        else:
            if (brightness < 0):
                brightness = 0
        if (brightness > 1):
            brightness = 1
        self.brightness_value = brightness

    # breathing light.
    def breath(self, R, G, B):
        breathSteps = 10
        for i in range(1,breathSteps):
            Breath_R = round(R*i/breathSteps)
            Breath_G = round(G*i/breathSteps)
            Breath_B = round(B*i/breathSteps)
            self.set_color_all(Breath_R, Breath_G, Breath_B)

            #self.pixels_show()
            #self.setColor(self.colorBreathR*i/self.breathSteps,
self.colorBreathG*i/self.breathSteps, self.colorBreathB*i/self.breathSteps)
            time.sleep(0.06)
        for i in range(1,breathSteps):
            Breath_R = round(R-R*i/breathSteps)
            Breath_G = round(G-G*i/breathSteps)
            Breath_B = round(B-B*i/breathSteps)
```

```
                        self.set_color_all(Breath_R, Breath_G, Breath_B)
                        #self.pixels_show()

                        #self.setColor(self.colorBreathR-(self.colorBreathR*i/self.breathSteps),
self.colorBreathG-(self.colorBreathG*i/self.breathSteps),                    self.colorBreathB-
(self.colorBreathB*i/self.breathSteps))
                        time.sleep(0.06)

        # Lights off
        def stop(self):
                robot_light.set_color_all(0,0,0)

        def test_multiple_colors(self):
                BLACK = (0, 0, 0)
                RED = (255, 0, 0)
                YELLOW = (255, 150, 0)
                GREEN = (0, 255, 0)
                CYAN = (0, 255, 255)
                BLUE = (0, 0, 255)
                PURPLE = (180, 0, 255)
                WHITE = (255, 255, 255)
                COLORS = (BLACK, RED, YELLOW, GREEN, CYAN, BLUE, PURPLE,
WHITE)
                for color in COLORS:
                        self.pixels_fill(color)
                        #self.pixels_show()
                        time.sleep(2)


if __name__ == '__main__':

        robot_light = WS2812()
        try:
                while True:
                        # test color multiple colors.
                        robot_light.test_multiple_colors()

                        # set a single color.
                        #robot_light.set_color_all(255,0,0)

                        # test breath led.
                        #robot_light.breath(250,0,0)
```

```
            # test adjust the brightness of the lights.
            #robot_light.brightness(0.03)

    except KeyboardInterrupt:       # Stop the program, the light goes out.
        robot_light.stop()
```

6.7 What's Next?

THANK YOU for participating in this learning experience!

If you find errors, omissions or you have suggestions and/or questions about this Lesson, please feel free to contact us: cokoino@outlook.com

We will make every effort to make changes and correct errors as soon as feasibly possible and publish a revised version.

If you want to learn more about Arduino, Raspberry Pi, Smart Cars, Robotics and other interesting products in science and technology, please continue to visit our website. We will continue to launch fun, cost-effective, innovative and exciting products.

http://cokoino.com/

Thank you again for choosing Cokoino products.